# Trabajo Fin de Grado

## Desarrollo de un sistema para la conversión de imágenes de video PAL a imágenes HD

Alumno: Martin Bozhidarov Ivanov
Tutora: Consuelo Gonzalo Martín

# Índice de General

# 1. Introducción

En las ultimas décadas hemos sufrido un gran cambio en el modo, como en la calidad de Vida en el cual se debe a gran medida al avance tan grande que ha habido en el mundo tecnológico. Alguno de estos avances y en el cual tratara el proyecto son la codificaciones y formato de video.

En las décadas que llevamos de televisión en color hay dos formatos de video en los cuales han destacado sobre el resto uno que es el sistema de codificación analógico PAL ,que es el sistema de televisión Analógica que se utilizaba en toda Europa (Exceptuando Francia) y en la mayoría de la población mundial. Por otro lado tenemos el otro sistema de video que es el HD aunque el proyecto lleva 40 años existiendo he tomado una mayor importancia ahora con el cambio que se ha habido de pasar de una televisión analógica a una televisión digital.

En este proyecto se creara una herramienta capaz de transformar un video en Formato PAL que es un formato que tiene 720 pixeles de longitud y 576 pixeles de altura al formato de video HD que en su caso tiene las dimensiones 1920x 1080 pixeles de longitud y altura respectivamente.

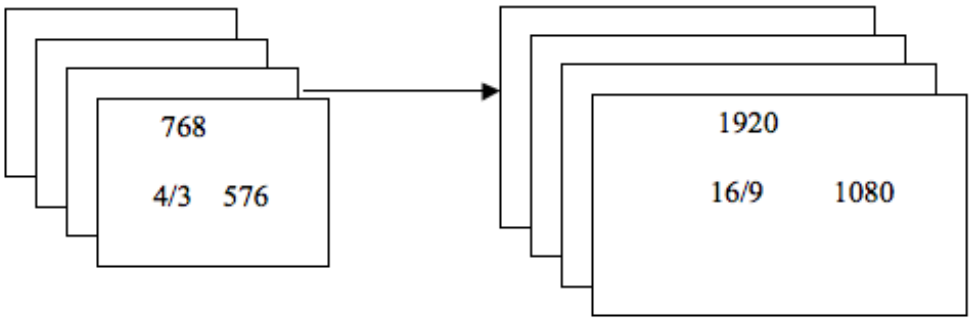



Fig. 1 Interpolación de la Imagen

Para realizar tal transformación debemos extraer todos los frames del video inicial para crear una imagen por cada frame (Fig. 1). Para realizar tal transformación se utilizara la herramienta ffmpeg en la cual dispone de múltiples opciones que nos servirán de gran ayuda. Una vez que tenemos las imágenes habrá que cambiarles de resolución para que se adapten al formato estándar de HD. Para tal cambio necesitamos los algoritmos de interpolación estos algoritmos sirven para rellenar los espacios que habrá entre los pixeles al aumentar la resolución de la imagen. Estos espacios o pixeles en blanco se rellenaran en función del algoritmo de interpolación que se utilice utilizando para ello los pixeles mas cercanos.

En este proyecto utilizaremos varios algoritmos para la obtención de imágenes pudiendo el usuario elegir entre algoritmos de interpolación que necesitan poco tiempo de computo pero obtienen imágenes difuminadas o otros algoritmos que requieren un tiempo de computo mayor pero obtienen imágenes nítidas.

Una vez obtenidas las imágenes en el formato final el programa deberá juntar las imágenes para obtener el video final para eso utilizaremos de nuevo la herramienta ffmpeg.

## 1.1. Motivaciones

A lo largo de la historia de la televisión de ha sufrido varios cambios que engloban las diferentes áreas de esta, pero sin duda el cambio mas importante ha sido el cambio de resolución de la emisión de la imagen al pasar de una imagen con resolución 4/3 a una resolución mas en forma rectangular que es 16/9, quedando los videos de la otra resolución obsoletos.

Los motivos por las cuales hemos decidido hacer este proyecto es poder disfrutar en pantalla completa con la mayor calidad posible de todos aquellos videos que forman parte de nuestra infancia o de nuestra juventud que pueden ser bautizos, bodas o series o películas favoritas que hemos ido coleccionando a lo lardo de aquella época.

Dado que no hemos sido capaces de encontrar una herramienta que ofrezca a los usuarios una forma rápida y sencilla de pasar sus antiguos videos que ha ido grabando en formato PAL convirtiéndolos a un formato mas actual como es el

formato HD perdiendo la menor Calidad posible tanto en el audio como en la visualización del Video, hemos decidido realizar este proyecto.

## 1.2. Objetivos

O1.- Determinar un algoritmo de interpolación que optimice la relación entre la calidad de las imágenes interpoladas y los recursos de cómputo requeridos.

O2.- Diseñar el sistema completo que a partir de una secuencia de imágenes ED genere la secuencia HD.

O3.-Implementar un prototipo del sistema que cumpla los requisitos establecidos

# 2. Estado de Arte

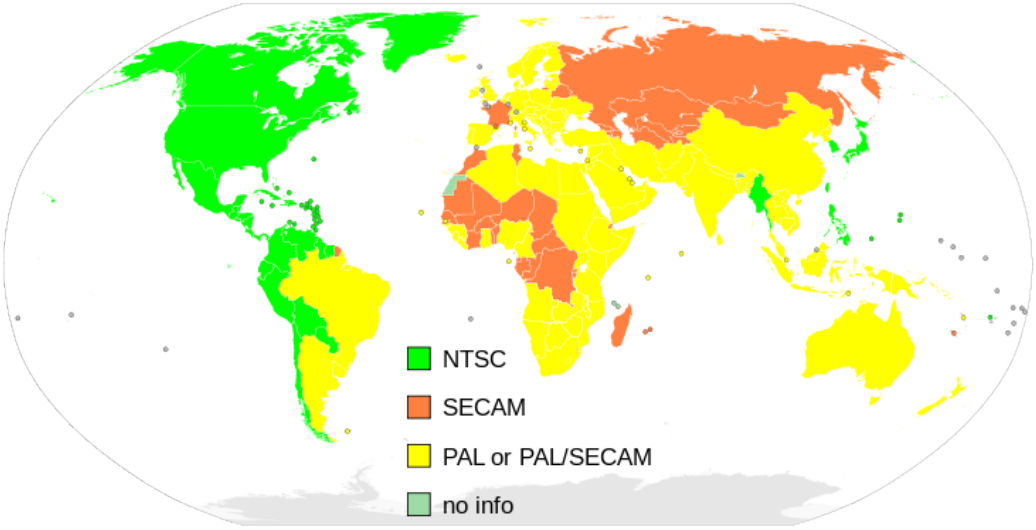
En esta parte del documento nos centraremos en el estudio monográfico, de una forma mas extensa de lo presentado en la parte de Introducción de este mismo documento.

## 2.1. Sistema de Video

### Video PAL

Es un formato de video analógico en el cual consta de 720 pixeles de largo y 576 pixeles de alto. Debemos destacar que estos pixeles para que cumplan el aspecto de radio de 4/3 deben ser pixeles rectangulares es decir que la dimensión del pixel es mayor de largo que de ancho por lo que a veces los sistemas PAL aparecen también con la resolución de 768x576 pixeles cuadráticos.
PAL (Phase Alternating Line) – es un sistema de televisión analógica de color, desarrollado por el ingeniero alemán de «Telefunken» Walter Bruch y adoptado como una televisión estándar en 1966 en Alemania, el Reino Unido y varios otros países de Europa occidental . Actualmente, el sistema PAL es más importante en el mundo Fig2.1. A finales de 1990, con este estándar de transmisión se utilizo en 62 países que era el 67,8% de los espectadores de todo el mundo.
Alguno de los aspectos del formato PAL:

- Relación de aspecto: 16:9
- Número de líneas: 625
- Líneas activas (resolución vertical efectiva): 576
- Columnas activas: 720
- Borrado vertical: 25 H + 12 microsegundos
- Frecuencia de cuadro: 25 Hz (40 ms)
- Frecuencia de campo: 50 Hz (20 ms, de los cuales 18,4 ms activos)
- Frecuencia horizontal o de líneas: 15,625 Hz
- Frecuencia de pulsos igualación: 31,250 Hz
- Frecuencia de la subportadora de crominancia: 4,4336 MHz (Modulada en amplitud y fase)
- Frecuencia de la señal P (PAL): 7,8 kHz (1/2 de la frecuencia de líneas)
- Periodo de línea (H): 64 $\mu$s
- Periodo activo de línea: 52 $\mu$s




Utilización del Sistema PAL en el mundo
Fig.2.1 Sistema Pal en el mundo

## Video HD


Es un formato de video digital en el cual consta de 1920 pixeles de largo y 1080 pixeles de alto utilizando de esta forma una relación de aspecto 16/9. El formato HD también tiene otras resoluciones que cumplen el aspecto de 16/9 que son 720x480 y 1280x720 pero nosotros en este proyecto utilizaremos el aspecto con mayor resolución que es 1920x1080.
HDTV (el formato HD utilizado en la televisión) tuvo sus inicios en el año 1972 ,aunque del termino alta definición ya se habla anteriormente, en este año se empezó hablar en una reunión la "Unión Internacional de Telecomunicaciones" con el fin de estandarizar el formato aunque no se llego a un acuerdo con lo que hubo varios aspectos de radio a lo largo de las siguientes décadas.
Mas tarde la relación de aspecto de 16/9 fue debidamente acordada en la primera reunión del grupo de trabajo IWP11 / 6 en Investigación de la BBC y el establecimiento Desarrollo en Kingswood Warren. La Recomendación UIT-R resultante UIT-R BT.709-2 (. "Rec 709") incluye la relación de aspecto de 16:9, una colorimetría especificado y los modos de escaneo 1080i (1080 líneas entrelazadas activamente en la resolución) y 1080p (1080 líneas escaneadas progresivamente).
Los ensayos Freeview HD británicos utilizaron MBAFF, que incluye contenido tanto progresivo y entrelazado con la misma codificación.

Nomenclatura del los videos HD:

| EBU System | Nomenclature [1] of abbreviations [activeLinesScanning /frame-rate] | Luma or R'G'B' Samples per active line (S/AL) | Active lines per frame (picture) (AL/F) | Frame rate, Hz | Luma or R'G'B' sampling [2] frequency (fs), MHz | Luma sample periods per total line (S/TL) | Total lines per frame | Net image Bit Rate (4:2:2, 10 bit) [GBit/s] | Corresponding SMPTE system nomenclature |
|---|---|---|---|---|---|---|---|---|---|
| S1 | 720p/50 | 1280 | 720 | 50 | 74.25 | 1980 | 750 | 0.92 | SMPTE 296M System 3 |
| S2 | 1080i/25 | 1920 | 1080 | 25 (50 Hz field rate) | 74.25 | 2640 | 1125 | 1.04 | SMPTE 274 System 6 |
| S3 | 1080p/25 | 1920 | 1080 | 25 | 74.25 | 2640 | 1125 | 1.04 | SMPTE 274 System 9 |
| S4 | 1080p/50 | 1920 | 1080 | 50 | 148.5 | 2640 | 1125 | 2.08 | SMPTE 274 System 3 |

Fig.2.2 Nomenclatura HD

## 2.2. FFMPEG


FFmpeg es un proyecto de software libre que crea bibliotecas y programas para el manejo de datos multimedia. FFmpeg incluye libavcodec, una biblioteca de codec de audio / vídeo que se utiliza en varios otros proyectos, libavformat, un contenedor de la biblioteca mux y demux audio / video y el programa de línea de comando ffmpeg para transformar archivos multimedia.
FFmpeg consta de los siguientes componentes :

- **ffmpeg** - utilidad de línea de comandos para convertir archivos de vídeo de un formato a otro. Con su ayuda , también puede capturar vídeo en tiempo real desde una tarjeta de TV .
- **ffserver** - HTTP ( RTSP se está desarrollando ) servidor de streaming para la transmisión de vídeo.
- ffplay - un reproductor basado en SDL y las bibliotecas FFmpeg .
- **libavcodec** - Biblioteca con todos los codecs de audio / video. La mayoría de los codecs se han desarrollado "desde cero" para garantizar el mejor rendimiento posible .

- **libavformat** - biblioteca con multiplexores y demultiplexores para varios formatos de audio y vídeo.
- **libavutil** - rutinas de biblioteca de apoyo a la norma común a los diversos componentes de ffmpeg . Incluye Adler- 32 , CRC , MD5 , SHA1 , LZO - descompresor Base64-koder/dekoder , DES, RC4 y AES
- **libpostproc** - una biblioteca de rutinas de procesamiento de vídeo estándar.
- **libswscale** - Biblioteca de escalado de vídeo .
- l**ibavfilter** – reemplazo de vhook, que le permite modificar la secuencia de vídeo entre el codificador y decodificador "sobre la marcha " .

## 2.3. Algoritmos de Interpolación

El algoritmo de interpolación es una o varias funciones matemáticas en las cuales se determina el valor de un punto desconocido a partir de los valores que tienen los puntos que están próximos a este.
En el proyecto utilizaremos estos algoritmos para pasar las imágenes de un formato inicial PAL de 720x576 pixeles a un formato final HD el cual tiene la resolución de 1920x1080 pixeles.

A lo largo de la historia se han definido muchos algoritmos de interpolación entre los cuales los hay con poco tiempo de computo pero no obtienen buenos resultados ya que difuminan mucho la imagen y por otro lado tenemos los algoritmos que requieren un mayor tiempo de computo pero en cambio obtienen unas resultados muy nítidos.

En este proyecto utilizaremos algoritmos de los dos tipos permitiendo al usuario elegir cual le conviene mas. Los algoritmos que utilizaremos en el proyecto son:

### 2.3.1. Mas Próximo

Este algoritmo es el primero que se utilizo y es el mas básico que se puede implementar a la hora de utilizar un algoritmo de interpolación ya que consiste básicamente en utilizar el pixel mas próximo con valor conocido para copiar este valor al pixel que en el cual desconocemos el error. Este algoritmo tiene un tiempo de computo muy bueno pero de unos resultados pésimos.

## 2.4. BiLineal

En matemáticas, la interpolación bilineal es una extensión de la interpolación lineal para interpolar funciones de dos variables (por ejemplo, x e y) en una cuadrícula regular 2D.

La idea clave es que para llevar a cabo la interpolación lineal primero se hace en una dirección, y luego en la otra dirección.

Un ejemplo de este algoritmo:

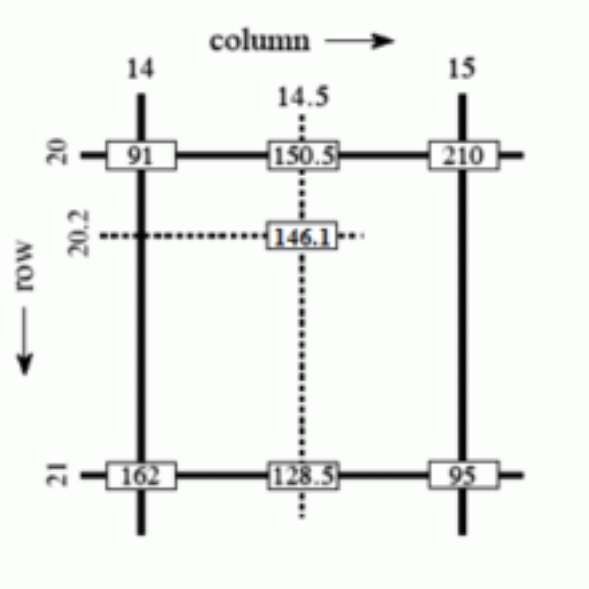



Fig2.3 Interpolación

En esta Fig2.3 podemos observar que el valor del centro (146,1) se obtiene a partir de los dos pixeles que están en el medio en la parte superior e inferior de la imagen dando vas valor al pixel que esta mas cerca. Estos dos pixeles a su vez estos dos se obtienen a parir de los pixeles de las esquinas de la imagen.

## 2.5. Bi-cubico

En matemáticas, interpolación bicúbica es una extensión de interpolación cúbica para la interpolación de puntos de datos en una cuadrícula regular de dos dimensiones o una imagen en nuestro caso. La superficie interpolada es más nítida que las superficies correspondientes obtenidos por interpolación bilineal o de la interolacion del vecino mas proximo. La interpolación bicúbica se puede lograr ya sea utilizando polinomios de Lagrange, splines cúbicos, o algoritmo de consolación cúbica.

En el procesamiento de imágenes, interpolación bicúbica se elige a menudo más que la interpolación bilineal en el remuestreo de la imagen, cuando la velocidad y el computo no es un problema. Mientras que la interpolación bilineal, que sólo tiene 4 píxeles (2x2) en cuenta, la interpolación bicúbica considera 16 píxeles (4x4)., obteniendo para eso mejores resultados.

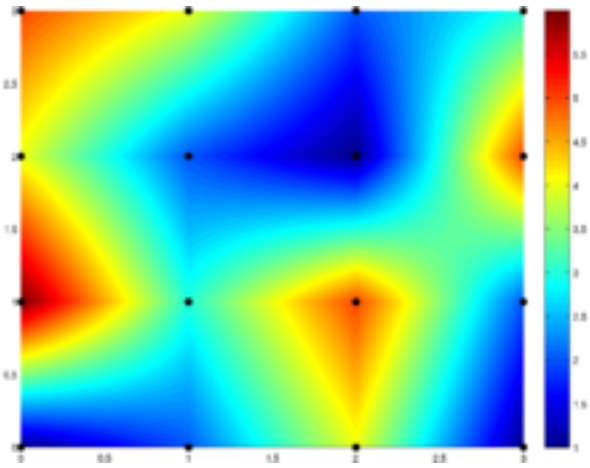


Fig.2.4 Ejemplo Interpolación lineal

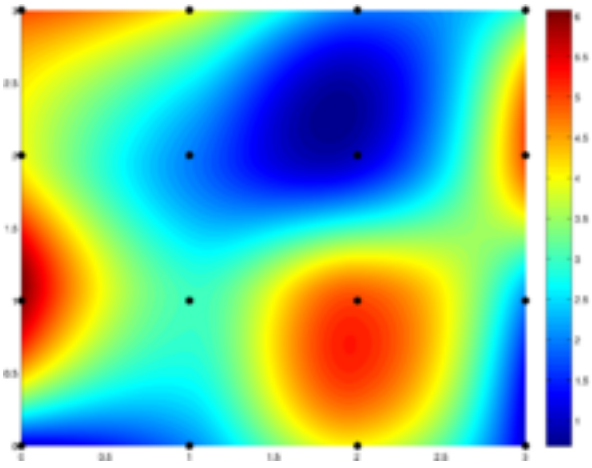


Fig2.5 Ejemplo Inter. bicubica

En estas dos imágenes podemos observar la mejora utilizando los algoritmos de interpolación Bi-cúbicos (Figura 2) respecto a los algoritmos de interpolación bilineales (Figura 1)

## 2.6. BMP

BitMap Pixel o mejor conocido por sus siglas "BMP" es un formato de imagen creado por Windows a mediados de los 90 para ser el formato de imagen que iba a utilizar el Sistema Operativo Windows. BMP utiliza los primeros 54 bytes para los metadatos o cabecera de la imagen estos datos son:

- Bytes del 0 al 1: Tipo de Fichero "BM"
- Bytes del 2 al 5: Tamaño del archivo
- Bytes del 6 al 9: Reservado
- Bytes del 10 al 13 : Inicio de los datos de la imagen
- Bytes del 14 al 17: Tamaño de la cabecera del bit
- Bytes del 18 al 21: Anchura (pixeles)
- Bytes del 22 al 25: Altura (pixeles)
- Bytes del 26 al 27: Numero de planos
- Bytes del 28 al 29: Tamaño de cada punto
- Bytes del 30 al 33: Compression (0= No comprimido)
- Bytes del 34 al 37 Tamaño de la Imagen
- Bytes del 38 al 41 Resolución Horizontal
- Bytes del 42 al 45 Resolución Vertical
- Bytes del 46 al 49 Tamaño de la tabla de Color
- Contador de colores importantes

A continuación de los metadatos sigue el cuerpo de la imagen es decir, valores de cada pixel. El primer pixel que aparece es el que esta situado a la esquina inferior izquierda siguiendo un orden de izquierda derecha y de abajo hacia arriba.

Debemos destacar que existen diferentes tipos de imágenes BMP pero en este proyecto utilizaremos 24 bytes por pixel que corresponden a los tres colores primarios de la luz que son Azul, Rojo y Verde cada uno de esos tres tendrá 8 bytes para representar un valor.

Los motivos por los cuales utilizaremos este formato de imagen es debido a que el propio formato trae la matriz de pixeles con lo que será mucho mas fácil manipular estos para realizar las operaciones de interpolación además nos ahorrarías transformar la imagen en mapa de bits con lo que conllevaría también perdida de calidad en las trasformaciones.

# 3. Implementación

En este capito mas vamos a ver todo el procedimiento que sufre el video desde que se introduce como un video PAL hasta que se genera el video correspondiente en un video en HD, excluyendo en este capitulo la parte de interpolación la cual se vera muy por encima ya que esta parte se expondrá con un mayor detalle en el siguiente capitulo.

Para poder realizar esta tarea de transformación hemos realizado la siguiente función:

principal(Entrada, Salida, algoritmo, compress, path );

De donde:

- Entrada: Es el video en Formato PAL el cual debemos convertir
- Salida: La ruta donde vamos a Guardar el video con el nombre que le queremos poner
- Algoritmo: El algoritmo que queremos implementar que pueden ser:
    - Mas Cercano
    - Lineal
    - Mitchell-Netravari
    - Bicubico Mejorado
- Compress: Si deseamos utilizar compression en el video final:
    - Compress==2: Compresion activada
    - Compress==1:Compresion desactivada
- Path: Es el directorio en el cual se encuentra el programa. Este valor lo coge automáticamente con lo que no hace falta especificarlo.

Una vez que llamamos al método, lo primero que debemos de hacer para poder tratar el video es generar una carpeta de temporales para generar esta carpeta en los sistemas operativos Linux y Mac se generara esta carpeta en el directorio "/tmp" y en el sistema operativo de Microsoft esta carpeta será "C:\\temp"

Para generar una carpeta aleatoria hemos decidido utilizar la función en C de srand en el cual nos generara un numero aleatoria en el que llamaremos asi a nuestra carpeta de esta forma si el programa se ejecuta dos veces esto no afectara al funcionamiento ya que no se mezclaran las imágenes y se podrá utilizar sin problemas de forma paralela. En nuestro cas hemos definido un intervalo de números que se puedan generar aleatoriamente de la siguiente manera:

```
srand(time(NULL));
numeroAleatorio = rand() %10000;
```

En los sistemas operativos de Mac y Linux al numero aleaorio le hemos añadido un punto y que de esta forma esta carpeta será una carpeta oculta quedando de la siguiente manera:

En Linux y Mac OS:

/tmp/.NumAleaorio

Windows:

C:\\Temp\NumAleatorio

Una vez que tenemos la carpeta de temporales generada nos tetemos empezar a procesar el video. Este procedimiento dependerá de que si se quiere utilizar compression o no .

## 3.1. Procesamiento sin Compresión

Teniendo en cuenta que querremos utilizar una herramienta que nos permita transformar nuestros videos PAL en un Formato en HD, de tal forma que durante esta transformación el video sufra la mínima perdida de calidad posible con lo que debemos utilizar el método de procesamiento sin compresión.

1. Lo primero que debemos hacer es generar todas las secuencias de imágenes del video dado que es un video pal se deberán de generara unas 25 imágenes por cada segundo que dura el video ,para eso vamos a utilizar el programa ffmpeg en el cual ejecutara el siguente comando:

ffmpeg –i VideoEntrada Imagenes

De Donde:

- VideoEntrada: Es nuestro video Pal que deseamos Convertir a HD

- Imágenes: Estas son las imágenes que se generaran a partir de los frames. Para poder organizar mejor las imágenes las imágenes pal tendrás el siguiente formato. /.NumAleatorio/%d.bmp donde %d es un numero natural.

2. Una vez que tenemos las imágenes correspondientes al Formato Pal vamos a utilizar la función de Interpolacion que se ha creado para tal caso que es:

   interpolacion(contador, nombreimagen, algoritmo, nombreCarpeta);

   Donde:

   - Contador : numero de imagen (o frame) de la imagen de esta forma se guardara la imagen en HD de forma ordenada
   - NombreImagen: Nombre de la Imagen que queremos interpolar
   - Algoritmo: Es el Algoritmo de Interpolación que estamos utilizando que puede ser:

     - Mas Cercano
     - Lineal
     - Mitchell-Netravari
     - Bicubico Mejorado

- Nombre Carpeta : La carpeta donde se van a guardar las imágenes en formato HD que tendrás la siguiente estructura HD%d.bmp donde %d es un numero natural.

3. Una vez que tenemos las imágenes en formato HD volvemos a utilizar la herramienta de ffmpeg para poder formar un video a partir de las imágenes que hemos creado. Para eso utilizaremos el siguiente comando:

   Ffmpeg –i Imágenes –vCodec Codec –y  Video2

   De donde:

   - Imágenes: Son las imágenes que hemos creado en el paso anterior en Formato HD la cuales tienen la siguiente estructura ./NumAleatorio/HD%d.bmp, de donde se cogerían todas las imágenes que cumplan ese patrón.

   - Codec: El códec que utilizaremos para el método sin compression es el códec v210 el cual junta todas las imágenes sin utilizar compression alguna ya que el video final ocupa lo mismo que la suma de todas las imágenes.

   - Video2: Es el video Final HD y vendrá con la ruta que le haya especificado el usuario para guardar el video

4. Una vez generado el video final debemos eliminar todas las imágenes que se hayan creado en la carpeta temporal incluido la carpeta temporal.

Este método esta muy bien ya que intenta utilizar reducir al máximo la perdida de calidad pero tiene un problema muy grande que es el espacio en disco que se puede generar a la hora de utilizar esta opción, ya que en nuestro ejemplo que era un video de 10 segundos el video generaba 250 imágenes (10 segundos x 25 frames(Imagenes)/segundo) teniendo en cuenta que una imagen en formato pal es de 1,2 MB y una imagen en formato HD es de 6 MB multiplicando las 250 imágenes de PAL y las 250 imagens de HD se generan alrededor de entre 1,5 GB y 2 GB que tratándose de un video de 10 segundos puede llegar a ser un verdadero problema si se trata de recodificar videos de mayor duración en un ambiente mas domestico.

Por lo que nos hemos visto obligados a crear otra forma de recodificar los videos que es la que se explicara acontinuacion que es el procesamiento con Compresion.

## 3.2. Procesamiento con Compresión

Dado el tamaño de las imágenes y sobre todo el tamaño del video final HD hemos querido añadir la opción que poder utilizar una compresión en el videos final ya que de esta forma la herramienta se pueda utilizar en un ambiente en el cual no se dispone de muchos recursos de almacenamiento como puede ser un ambiente domestico en los cuales las personas no se pueden permitir un consumo excesivo de disco duro.

La parte que se pueda contrarrestar de utilizar el método de compresión respecto al método sin compresión es la posible perdida de calidad en el video final al utilizar esta técnica, además que con la utilización de Compresión a la hora de la recodificación del video el rendimiento del ordenador es mas bajo debido a que este consume mas CPU ya que no sigue los pasos de recodificación que se utilizaban en la opción sin Compresión.

Tal como dijimos anteriormente nuestra intención con esta técnica es la utilización del mínimo espacio posible para la realización de la tarea con lo que este tarrea sigue los siguientes pasos:

1. Se establece un contador=0 de tiempo que ira aumentando en cada iteración cada 4 Segundos

2. Se intrudice el siguiente comando:

   ./ffmpeg -ss Contador –i Video_PAL –v:frames N Imagenes

   - Contador: Es el tiempo expresado en horas, minutos y segundos que tiene el siguiente formato HH:MM:SS

   - Video_PAL : Es el video PAL de entrada

   - N: es el numero de frames. Dado que hacemos videos de 4 segundos y los frames en los videos PAL son de 25 Frames/s, el valor que le debemos poner a N es 25x4=100 frames

   - Imágenes : Son los imágenes que se van generando a partir de los frames normalmente en el proyecto se utiliza este nomenclatura para nombrar las imágenes, /.NumAL/%d.bmp donde NumAL es un numero aleatorio y %d es la numeración que tendría cada imagen.

3. Una vez que tenemos las imágenes generadas, cada una de estas imágenes pasan a la parte de interpolación en la cual según el algoritmo especificado tardara mas o menos tiempo en terminar de generar todos las imágenes. Para la generación de las imágenes utilizamos el siguiente metodo:

   interpolacion(contador,nombreimagen,algoritmo,nombreCarpeta);

   Donde:

   - Contador : numero de imagen (o frame) de la imagen de esta forma se guardara la imagen en HD de forma ordenada
   - NombreImagen: Nombre de la Imagen que queremos interpolar
   - Algoritmo: Es el Algoritmo de Interpolación que estamos utilizando que puede ser:

     - Mas Cercano
     - Lineal
     - Mitchell-Netravari
     - Bicubico Mejorado

   - Nombre Carpeta : La carpeta donde se van a guardar las imágenes en formato HD que tendras la siguiente estructura HD%d.bmp donde %d es un numero natural.

4. Una vez que tenemos todas las imágenes en HD vamos a proceder a juntar todas las imágenes en un video para este proceso necesitaremos otra vez de ffmpeg con lo que el programa ejecutar el siguiente comando:

   ffmpeg –i Imágenes –vCodec Codec –y Video2 (Video1)

   De donde:

   - Imágenes: son la Imágenes en formato HD que hemos interpolado que tienen a siguiente nomenclatura ./NumAl/HD%d.bmp , de donde NumAl es un numero aleatorio y %d es un numero natural.

   - Codec: es el códec por con el cual vamos a crear el video en nuestro caso será “libx264”

   - Video2: es nuestro video final de salida. En el caso en el cual esta operación forma parte de la primera iteración Video2 se llamara Video1

5. Tal dijimos este procedimiento se hará de 4 en 4 segundos con lo que si este paso no es el primero en hacerse debemos concatenar este video con el anterior creado, con lo que si esta es la primera iteración este paso se excluye.
   Para generar el video vamos a utilizar de nuevo ffmpeg con la opción que no ofrece el programa para concatenar videos, el comando por el cual vamos hacer la concatenación es el siguiente:

   ffmpeg -i 'concat:Video1|Video2\' -vcodec Codec -c copy -y Video3

   De donde:

   - Video1: Video en el cual están concatenados todos los anteriores videos generados.

   - Video2 : Video generado en esta iteración

   - Codec: es el códec por con el cual vamos a crear el video en nuestro caso será “libx264”

   - Video3: Video final que contiene el contenido del Video1 y del Video2 Concatenado

6. Ahora debemos comprobar hay mas video que debemos trasformar con lo que para eso analizamos el numero de ficheros que hay en la carpeta(al ser una carpeta que hemos generado aleatoriamente en esta carpeta solamente debe haber ficheros nuestros). Para este caso debemos tener en cuenta dos opciones:

   - Primera Iteración: En esta primera iteración si hay menos de 203 ficheros eso significa que el video en cuestión dura menos de 4 segundos ya que en 4 segundos se generan 100 imágenes en formato PAL + 100 imágenes en formato HD + el video final además de las carpetas “.” y “..” esto suma un total de 203 ficheros

   - Segunda iteración en adelante: En las Segunda iteración en adelante ahora además de los 203 ficheros que tenemos hay dos ficheros mas que son en nuestro caso el video final concatenado (Video3) y el video origen(Video1).Por lo que esta vez debemos comprobar si hay menos de 205 ficheros para salir de la iteración

7. Una vez comprobados los ficheros vamos a proceder a borrar todas las imágenes que hemos ido generando en esta iteración.

8. Si se da el vaso en el cual no se va a salir del bucle renombremos el Video3 a Video1 , borrando de esta forma todo el contenido del Video1 ya que se ha quedado obsoleta la información debido a que lo que nos interesa es mantener un video que tenga el contenido desde el principio hasta la actual iteración.

9. Borramos todas las imágenes. En el caso de que se requiere otra operación se vuelve al apartado 1.

10. Si ya no hace falta otra iteración lo que debemos hacer es transforma el video final al formato de salida y en la carpeta donde se querria guardar el fichero para esto utilizaremos el siguiente comando:

    ffmpeg –i Video3 –vCodec Codec –y VideoSalida

    de donde:

    - Video3: Video final con formato especial
    - Codec: es el códec por con el cual vamos a crear el video en nuestro caso será “libx264”
    - VideoSalida: Es el video final que se guardara en la ruta especificada por el usuario y con el formato deseado.

11. Por ultimo y una vez completada la fase 10 borramos todos los temporales incluyendo la carpeta que hemos generado.

Por motivos de formato se decidió hacer las videos intermedios en formato .AVI debido a que muchos formatos incluyendo entre ellos el formato .MOV incluyen una cabecera al principio de cada video con lo que la concatenación no se puede hacer con un simple paso como es el caso del formato .AVI.

El tamaño de los temporales dependerá sobre todo del tamaño del video generado pero mas o menos el tamaño será de 600 MB (1 x 100 imágenes en HD) + 150MB (1 x 100 Imágenes Pal) + Video1 y Video3, que no sabemos el tamaño exacto debido a que en cada video dependiendo de la duración puede variar el tamaño de los temporales.

## 3.3. Utilización de Codecs

Para poder reproducir correctamente los videos finales debemos tener instalados en nuestro equipo los siguientes codecs:

- V210: es el códec que se utiliza para la reconfiguración del Video a HD sin la utilización de Compresión

- x264: Este es el códec que utilizamos para reproducir los videos en HD que hayan utilizado la opción de compresión de datos.

# 4. Interpolación de Imágenes

En el capitulo anterior nos centramos en todo el proceso de la separación de las imágenes a partir de un Video Pal y posteriormente la unión de todas las imágenes generadas a partir de las interpolación, quedando este proceso sin mencionar

En este capitulo nos vamos a centrar en todo el proceso de interpolación de las imágenes en las cuales a partir de una resolución de 720x576 pasan a una resolución de 1920x1080 de las cuales se utilizaran estos algoritmos de interpolación:

- Mas Cercano
- Lineal
- Mitchell-Netravari
- Bicubico Mejorado

Estos algoritmos se explicaran con una profundidad a lo largo de este capitulo en el cual tendrán un apartado cada uno.

Para la generación de las imágenes interpoladas utilizamos la funcicion de interpolación la cual explicamos sus argumentos en el apartado anterior:

interpolacion(contador, nombreimagen, algoritmo, nombreCarpeta);

Una vez recibidos todos los parámetros y ya dentro del método necesitamos abrir la imagen y leer de ella los datos lo primero que haremos es crear una estructura para poder leer correctamente la cual tiene la siguiente estructura:

```
{

WORD   identificadoe;       // Magic number: BM
DWORD  tamano;              // tamaño del archivo
DWORD  reservado;           // reservado
DWORD  offset;                 // offset para comenzar a leer la data
DWORD  tamanoCab;        // tamaño de la cabecera
DWORD  ancho;               // ancho
DWORD  alto;                // alto
WORD   planos;              // numero de planos del bitmap
WORD   bitsPorPixel;       // bit por pixel
DWORD  compresion;      // especificacion de la compresion
DWORD  imageTam;        // tamaño de la seccion de datos
DWORD  horizontalRes;   // resolucion horizontal  medida pixeles x metro
DWORD  vresolution;       // resolucion vertical medida pixeles x metro
DWORD  numberOfColours;  // numero de colores ( 2^bitsPerPixel )
DWORD  importantColours;

}BMPCabecera;
```

Esta parte es muy importante leerla debido a que a la hora de guardar la imagen en HD debemos modificar varios campos de esta cabecera.

Además de la estructura de la cabecera debemos crear otra estructura la cual nos servirá para poder manejar de forma mas fácil los pixeles. Debido a que cada pixel tiene tres colores de 1 Byte cada una hemos creado la siguiente estructura:

```
typedef struct
{
BYTE b;  // color azul
BYTE g;  // color verde
BYTE r;  // color rojo
}Color;
```

Una vez que tenemos la cabecera leída necesitamos leer y escribir en los pixeles correspondientes. Para poder manipular los pixeles en las imágenes de una forma mas sencilla hemos credo los siguientes métodos:

- GetPixel: Este método recibe como parámetro las coordenada y las abscisas de la imagen además de la imagen de la cual queremos leer los datos y nos devuelve la estructura Color con el valor de los tres colores en este punto. Dado que solamente se utiliza este método para la imagen en Formato PAL cualquier calor de las coordenadas de sea superior a

720 o inferior a 1 o que en las abscisas hay un valor superior a 576 o inferor a 1 lanzara un error

- SetPixel: Este método recibe como parámetro las coordenadas y las abscisas del pixel además de los valores de los colores correspondientes al pixel y la imagen en la cual queremos escribir. Dado que solamente se utiliza este método para la imagen en Formato HD cualquier calor de las coordenadas de sea superior a 1920 o inferior a 1 o que en las abscisas hay un valor superior a 1080 o inferor a 1 lanzara un error. Este método devuelve un int.

Para las interpolaciones bicubicas que necesitan 16 pixeles hemos creado otro método que se llama GetPixel16, esta forma no llamar 16 veces a Getpixel sino con una llamada coger todos los pixeles que necesitamos.:

- GetPixel16 : Este método tiene los mismos argumentos pero devuelve un array de color que contendrá los valores de cada pixel.

| | | | |
|---|---|---|---|
| 13 | 14 | 15 | 16 |
| 9 | 10 | 11 | 12 |
| 5 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 |

Fig4.1 Tabla Pixeles

Esto es lo que nos devolverá el array en este caso la celda 6 es el pixel principal(el que pasamos como argumento), y nos devuelve los otros quince pixeles mas cercanos a este.

Una vez definidas las funciones debemos proseguir la ejecución de los algoritmos de interpolación

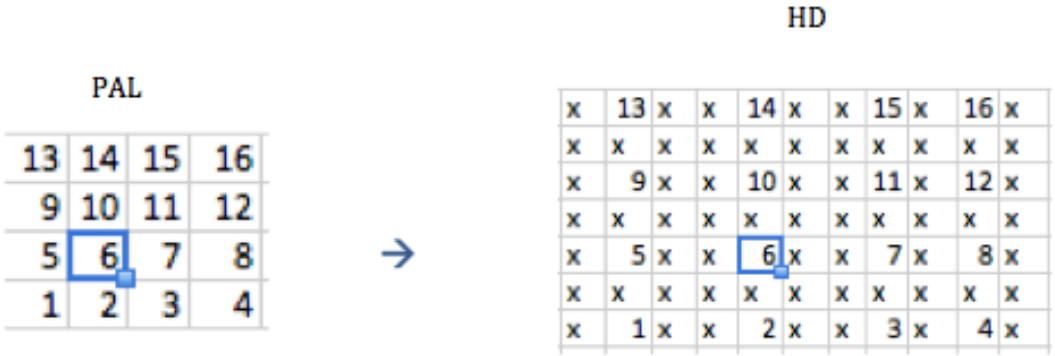



Fig.4.2 Colocación de bits

En esta Fig.4.2 se muestra un pequeño ejemplo de cómo quedarían los pixeles correspondientes en la imagen PAL en la imagen correspondiente en HD quedando los pixeles x como incognitos en los cuales intentaremos rellenar utilizando los diferentes tipos de algoritmos de interpolación.

## 4.1. Interpolación Vecino Mas Próximo

Tal como dijimos en el apartado del Estudio monográfico este es el algoritmo mas básico que se puede utilizar el cual consiste en copiar el valor del pixel mas próximo del cual conocemos el valor al pixel que queremos interpolar.

Para implementar este algoritmo creamos dos constantes :

- ConstHor: constante que viene definida por 720(pixeles horizontales en PAL)/1920(pixeles horizontales en HD)
- ConstVer: constante que viene definida por 576(pixeles verticales en PAL)/1080(pixeles verticales en HD)

Una vez definidas las dos constantes recorremos la matriz HD y para cada valor X e Y la multiplicamos por su valor de constante correspondiente es decir:

- X1=(int)X/ConstHor
- Y1=(Int)Y/ConstVer

Donde X e Y son los calores de la imagen HD e X1 e Y1 obtendrá de imagen PAL .Con esto y con el truncamiento que se hace al pasar a valores enteros obtenemos el valor mas cercano. Para los casos en los cuales el truncamiento es igual a 0 para alguno de las variables X1 e Y1 modificamos la variable correspondiente al valor 1.

## 4.2. Interpolación Lineal

Esta táctica consiste básicamente en coger los cuatro pixeles mas cernamos al pixel interpolado y realizar

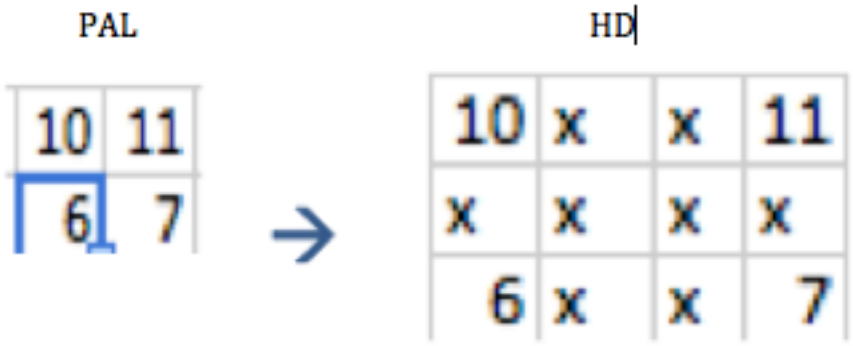



Fig.4.3 Colocación de Bits Alg. Lineal

Para la interpolación lineal el procedimiento que seguiremos es ir cuadricula por cuadricula tal

Tal como hablamos como se ve en la Figura 1.

Primero que debemos hacer es ubicar el pixel( pixel 6 (Fig4.3)) de PAL en HD tal como hicimos en la parte de la interpolación del bit mas cercano. Una vez tenemos la ubicación del pixel en HD hacemos lo mismo para los pixeles de las esquinas del cuadrante (pixeles 7,10,11 (Fig4.3))

Una vez que tenemos los pixeles calculamos las distancias entre el pixel 6 y pixel 7 y la distancia entre el pixel 6 y 10 una vez que tenemos las dos distancias distHor y DistVer que son distancias horizontal y distancia vertical consecutivamente. Invertimos el valor:

- inHor= (1/DistHor)
- inVer=(1/DistVer)

Las distancias de un pixel desconocido se calcula de la siguiente manera posición pixel – posición pixel 6 las distancias la llamaremos (h,v) donde h es la distancia vertical y v la distancia horizontal estas dos distancias las multiplicaremos por inHor e incver respectivamente con lo que obtenemos a para las coordenada y b para la abscisas.

X1=((1-b)*((1-a)*pixel6 + pixel7*a))+ b*((1-a)*pixel10 + a*Pixel11)

Los pixeles son los correspondientes a la Fig.4.3.

Pala los pixeles en los cuales no hay cuadricula se utiliza la interpolación del bit mas cercano que es solamente en la columna 1 de la imagen.

## 4.3. Interpolación Mitchell-Netravari

Para llevar a cabo esta interpolación utilizáremos para los 16 pixeles mas cercanos para poder interpolar mejor el pixel.

Para pasar de la coordenada de PAL a la coordenada HD utilizaremos el mismo procedimiento que hemos ido utilizando para el caso lineal y el caso del vecino mas cercano multiplicando las coordenadas y las abscisas por ConstHor y ConstVer. De esta forma Ubicamos en la imagen los 16 pixeles además para asignarles el valor utilizamos el método getpixel16 para rellenar los valores correspondientes en la imagen HD como se muestra a continuación en la imagen

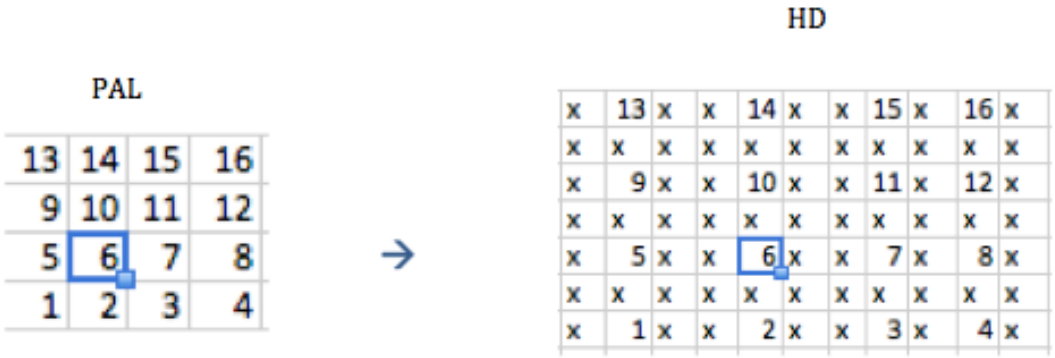



Fig.4.4 Transformación Pixeles Mitchell-Netravari

Durante esta iteración solamente se rellenaran los valores que forman el siguiente cuadrado:

| | | | |
|---|---|---|---|
| 10 | x | x | 11 |
| x | x | x | x |
| 6 | x | x | 7 |

Fig.4.5 Matrix central Mitchell-Netravari

Una vez que tenemos todos los pixeles vamos a recorrer la matriz (Fig.4.5) para eso necesitamos las variables a y b que se obtienen de la misma forma que en el apartado anterior en la interpolación lineal. Una vez que tenemos eso lanzamos el siguiente bucle:

```
for (n=-1;n<=2;n++){
  for(m=-1;m<=2;m++){

    decimalhor=(b-m);
    coef1=CatMullRom(decimalhor);

    decimalver=(a-n);
    coef2=CatMullRom(-decimalver);
    C1=C1 + (pixel[s].g*coef1*coef2);
    C2=C2 + (pixel[s].r*coef1*coef2);
    C3=C3 + (pixel[s].b*coef1*coef2);

    C4=C4 + (coef1*coef2);

    s++;
  }
}
```

Esto se hace para cada color en el que consiste en que :

- C1 guarda todos las sumas de la multiplicación de los dos coefeicientes además del valor del pixel verde
- C2 guarda todos las sumas de la multiplicación de los dos coefeicientes además del valor del pixel rojo
- C3 guarda todos las sumas de la multiplicación de los dos coefeicientes además del valor del pixel azul
- C4 guarda la suma de la multiplicación de cada uno de los dos coeficientes

Donde pixel [s] es el array de los 16 pixeles CatMullRom es el método que llamamos para la interpolación que es el siguiente(Fig.4.6)

$$k(x) = \frac{1}{6}\begin{cases}(12-9B-6C)|x|^3 + (-18+12B+6C)|x|^2 + (6-2B) & \text{if } |x|<1 \\ (-B-6C)|x|^3 + (6B+30C)|x|^2 + (-12B-48C)|x| + (8B+24C) & \text{if } 1\le|x|<2 \\ 0 & \text{otherwise}\end{cases}$$

Fig.4.6 Algoritmo Interpolación Mitchell-Netravari

Una vez que tenemos los valores C1,C2,C3 y C4 hacemos lo siguiente:

```
pix= (int) C1/C4;
  pixelRes.g=pix;
  if (pix<0) {
     pixelRes.g=0;
  }
  if (pix>255) {
     pixelRes.g=255;
  }
```

Para cada color utilizamos la C que corresponda.

Por ultimo se comprueba que el bite no exceda el intervalo(x>=0 y x<=255). Por ultimo se utiliza el método Set pixel para guardar el dato en la imagen Resultante .

Para los pixeles como frontales de la imagen donde no hay 16 pixeles contiguos se utiliza la interpolación lineal.

Por ultimo Guardamos la imagen con los valores de la cabecera correspondientes.

## 4.4. Interpolación Bicubico Mejorado

Para la interpolación del bicubico Mejorado el interpolado como también como en el caso del algoritmo de Mitchell-Netravali se utilizaran los 16 pixeles mas cercanos para poder acceder a ellos utilizamos el mismo procedimiento que en el caso anterior.

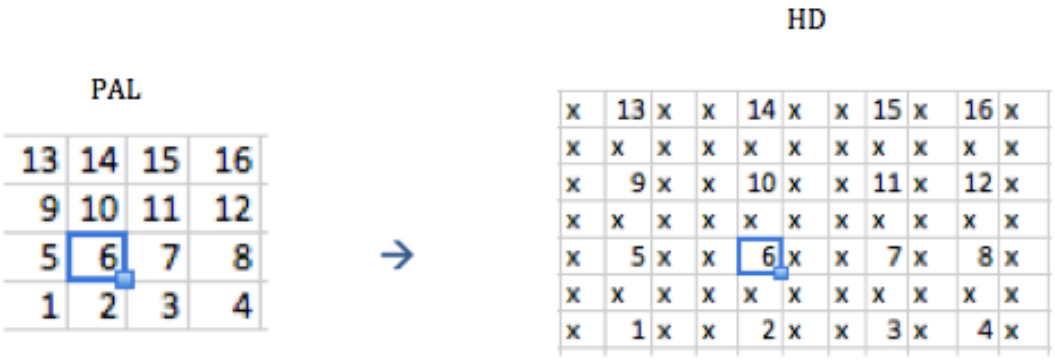



Fig.4.7 Transformación Pixeles Bicubico mejorado

Tal como paso en el algoritmo bicubico en este caso solamente se van a procesar el siguiente cuadro(Fig.4.8)

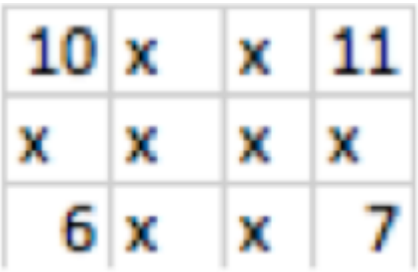



Fig.4.8 Matriz central Bicubico Mejorado

Una vez que tenemos el cuadra definido vamos a empezar a recorrerlo y escribir en el los datos que vamos interpolando la función de interpolación es la siguiente:

```
float *interpoladomej ( float diferencial, float ConstA){
  //  float valor;
   static float C[4];
    C[0] = -a*pow(diferencial,3) + a*pow(diferencial,2);
    C[1] = -(a+2)*pow(diferencial,3) + (2*a +3)*pow(diferencial,2) -
(a*diferencial);
    C[2] = ((a+2)*pow(diferencial,3)) - ((a+3)*pow(diferencial,2)) +
1;
    C[3]  =  (a*pow(diferencial,3))  -  (2*a*pow(diferencial,2))  +
(a*diferencial);

    //valor = (int) (C4*t + C3*s +C2*y + C1*z);

    return C;

}
```

donde:

- ConstA es la Constante A que puede tener un valor entre 1 y -1
- Diferencial es la distancia horizontal o vertical respecto al pixel 6.

Esta función nos devuelve un array de valores.

La primer vez que ejecutamos esta función en el diferencial ponemos la distancia verttical con lo que obtenemos el array de 4 variables float. Lo que devuel lo llamaremos CveCher[4]

Ahora lo que debemos comprobar si el pixel que queremos interpolar es mas cercano al pixel6 o mas cercano al pixel 7 si el pixel en concreto es mas cercano al 7 con lo que :

En el caso que el pixel sea mas cercano al pixel 6 hacemos:

```
C1 =  (cvechor[3]*pixel[1].g + cvechor[2]*pixel[2].g +cvechor[1]*pixel[3].g +
cvechor[0]*pixel[4].g);
C2 =  (cvechor[3]*pixel[5].g + cvechor[2]*pixel[6].g +cvechor[1]*pixel[7].g +
cvechor[0]*pixel[8].g);
```

```
C3 =  (cvechor[3]*pixel[9].g + cvechor[2]*pixel[10].g +cvechor[1]*pixel[11].g + cvechor[0]*pixel[12].g);
C4 =  (cvechor[3]*pixel[13].g + cvechor[2]*pixel[14].g +cvechor[1]*pixel[15].g + cvechor[0]*pixel[16].g);
```

En cambio si este es mas cercano al pixel 7 hacemos:

```
C1 =  (cvechor[3]*pixel[4].g + cvechor[2]*pixel[3].g +cvechor[1]*pixel[2].g + cvechor[0]*pixel[1].g);
C2 =  (cvechor[3]*pixel[8].g + cvechor[2]*pixel[7].g +cvechor[1]*pixel[6].g + cvechor[0]*pixel[5].g);
C3 =  (cvechor[3]*pixel[12].g + cvechor[2]*pixel[11].g +cvechor[1]*pixel[10].g + cvechor[0]*pixel[9].g);
C4 =  (cvechor[3]*pixel[16].g + cvechor[2]*pixel[15].g +cvechor[1]*pixel[14].g + cvechor[0]*pixel[13].g);
```

La diferencia entre los dos métodos es la ordenación de los pixeles horizontalmente ya que Chevor[3] y cvehor[2] tiene un mayor valor que Chevor[0] y chevor[1] respectivamente con lo que es lógico que los pixeles que sean mas cercanos al pixel interpolado tengan mayor coeficiente a la hora de la interpolación.

Por ultimo utilizamos las el array C[4] y chever para optener el pixel:

```
pix= (int) (C1*cvecver[3] + C2*cvecver[2] +C2*cvecver[1] +C2*cvecver[0] );
```

```
pixelRes.g=pix;
                    if (pix<0) {
                       pixelRes.g=0;
                    }
                    if (pix>255) {
                       pixelRes.g=255;
                    }
```

Al final comprobamos si estamos el valor cumple el intervalo establecido. Este procedimiento se hace para todos los colores.

# 5. Comparación de Resultados

En esta sección vamos a realizar un estudio con los resultados obtenidos con los diferentes algoritmos de interpolación.

Para un mejor estudio de los resultados hemos partido de un fragmento de video en definición HD y lo hemos pasado a un video PAL de esta forma podemos observar de mejor forma la perdida de calidad que puede sufrir la imagen dependiendo del algoritmo utilizado.

Para la medición del tiempo de cumputo para el apartado del tiempo de computo estas pruebas se han reailizado sobre un Mac BookPro 13" 4GB de memoria RAM y un procesador 2.4 GHz Intel Core i5

Lo primero que haremos es una exposición de una imagen en cada uno de los algoritmos que en esta imagen haremos varios capturas aumentanto el zoom para que se pueda ver de una forma mas visible los pro y los contra de cada imagen

## 5.1. Imagen HD original

Partiendo de la imagen original en HD que es la siguiente

Fig.5.1 Imagen Original zoom 100%

Imagen aumetada al 300%

Fig.5.2 Imagen Original zoom 300%

Por ultimo aumentamos aun mas la imagen al 600 % para analizar aun mas los tetalles

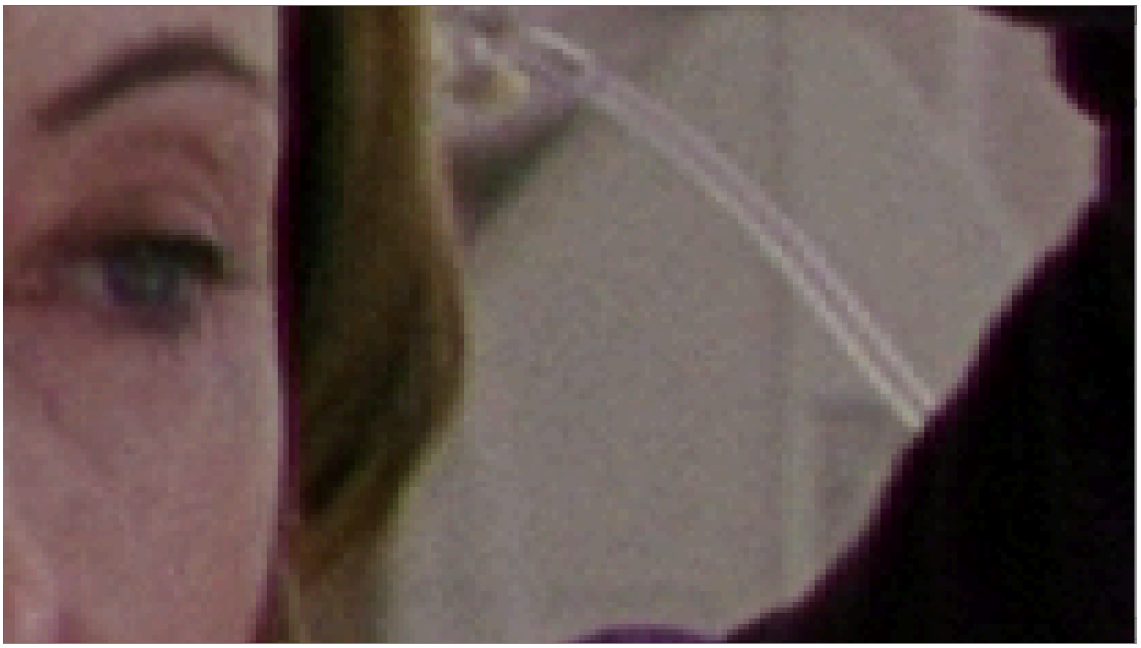


Fig.5.3 Imagen Original zoom 600%

Fig.5.4 Imagen Original zoom 600% Sabana

En esta imagen (Fig.5.4) podemos observar una especie de pétalos de flores dibujadas en las sabanas.

## 5.2. Vecino mas Próximo

### Calidad de Imagen

Fig.5.5 Imagen vecino mas proximo zoom 100%

En esta captura (Fig.5.5)es muy difícil que se vea la perdida de calidad con lo que vamos a hacerles un zoom al 300 % para poder mejor ver y los resultados (Fig.5.6).

Fig.5.6 Imagen HD 300% utilizando interpolación vecino mas próximo

En esta imagen ya se empieza a notar la perdida de calidad al empezar a estar muy pudelada la imagen

Por por ultimo vamos a acercar las imágenes al 600% de para poder analizar mejor las la imagen(Fig.5.7).

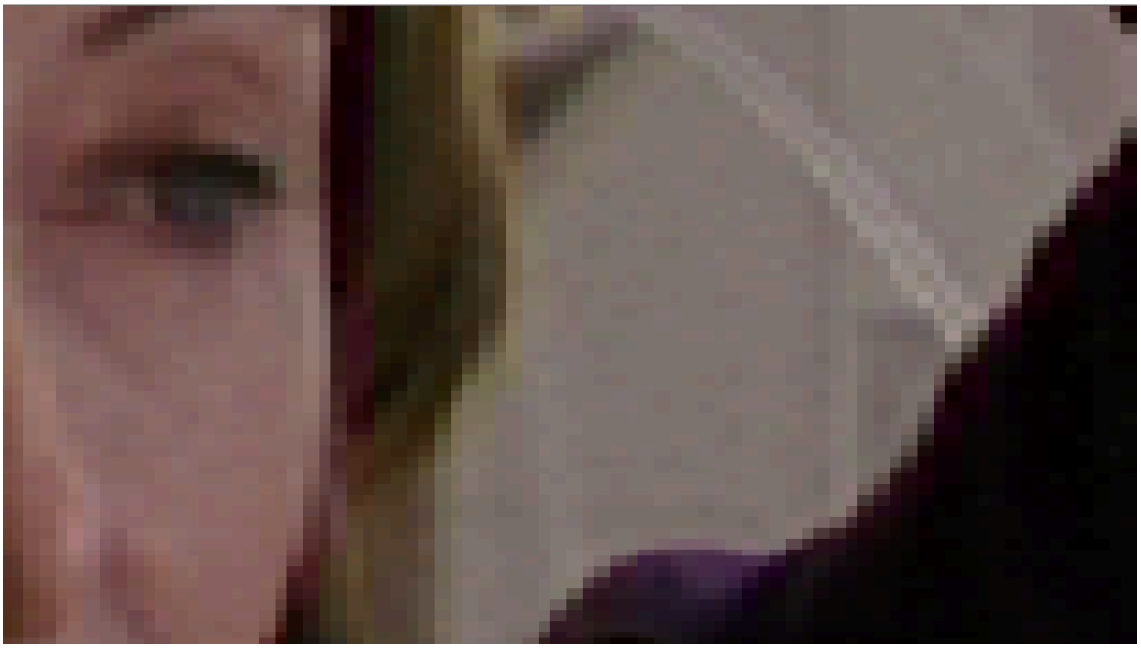


Fig.5.7 Imagen HD 600% utilizando interpolación vecino mas próximo

Como podemos observar la tanto los detalles de los ojos como por ejemplo los tubos (ya que ahora es mas difícil distingue entre los dos tubos) han perdido bastante su claridad además de que la imagen es parece mucho mas pudelada que su imagen homologa en HD .

Otras partes de la imagen acercadas al 600% (Fig.5.8)

Fig.5.8 Imagen HD 600% vecino mas próximo, Sabana
Otro claro ejemplo (Fig.5.8) en el cual se ver que en esta interpolación ya se han perdido los detalles de los pétalos de flores en las sabanas.

## Tiempo te computo

La parte positiva a la hora de utilizar este método es que el tiempo de computo es mucho mas corto que en los demos algoritmos el cual es capaz de generar las 250 imágenes en menos de 2 Minutos quedando de esta forma una media de 2 segundos por imagen trasformada con este algoritmo

## 5.3. Interpolación Lineal

### Calidad de la Imagen

Fig.5.9 HD con interpolación lineal 100%

En esta imagen (Fig.5.9) no podemos observar una perdida de calidad

Fig.5.10 HD interpolación lineal acercada al 300%

A partir de esta imagen (Fig.5.10) ya se empieza a notar e reducir la calidad con respecto
A la imagen original HD pero para ver con mas detalle aun las diferencias aumentamos la imagen al 600% (Fig.5.11)

HD interpolación lineal acercada al 600%

Fig.5.11 HD interpolación lineal acercada al 600%

otra toma  al zoon de 600% (Fig5.12)

Fig.5.12 HD interpolación lineal acercada al 600% en sabana

En esta imagen (Fig.5.12) las pétalos de flores todavía no son reconocibles pero por lo menos se puede observar una especie de círculos y dentro de el un punto,

por lo que esta técnica obtiene unos mejores resultados que la técnica del vecino mas próximo

### Tiempo te computo

El tiempo de computo de las imágenes lineales puede llegar a una media de 250 imágenes en 3 minutos con lo que estos nos da unos computo de 1,4 Imágenes por Segundo

## 5.4. Mitchell-Netravali

### Calidad de la Imagen

Imagen original (Fig5.13)

Fig.5.13 Imagen interpolación Michael_Netravali acercada al 100%

Imagen reducida al 300% (Fig.5.14)

Fig.5.14 Imagen interpolación Michael_Netravali acercada al 300%

Imagen aumentada al 600% (Fig.5.15)

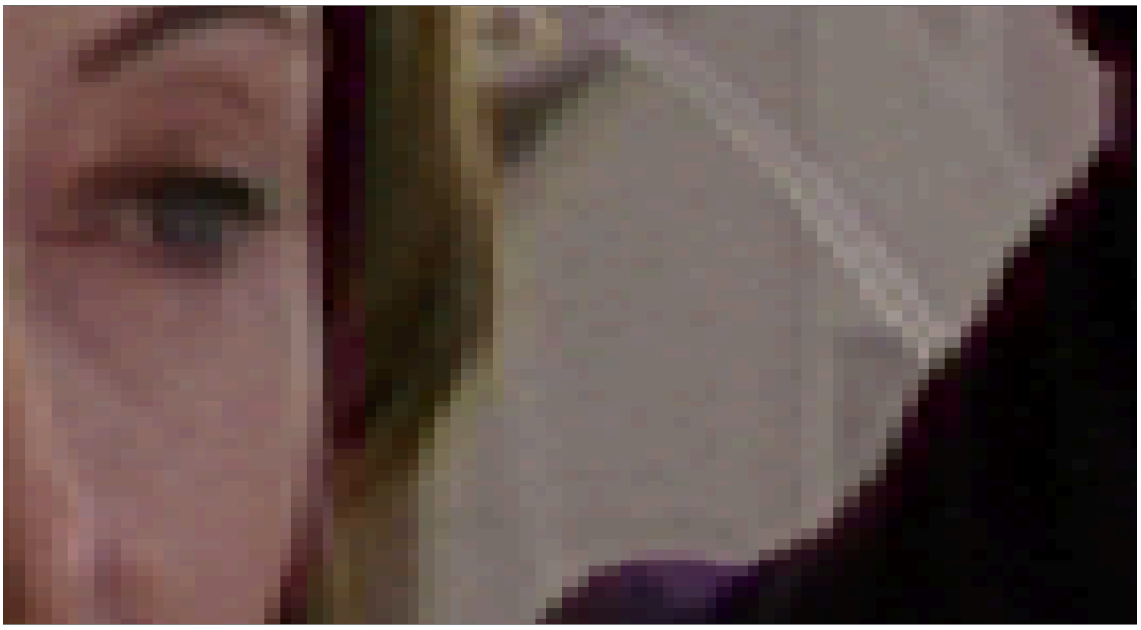


Fig.5.15 Imagen interpolación Michael_Netravali acercada al 600%

Otra toma te la imagen con la interpolación de Mitchell-Netravali aumentada al 600% (Fig.5.16)

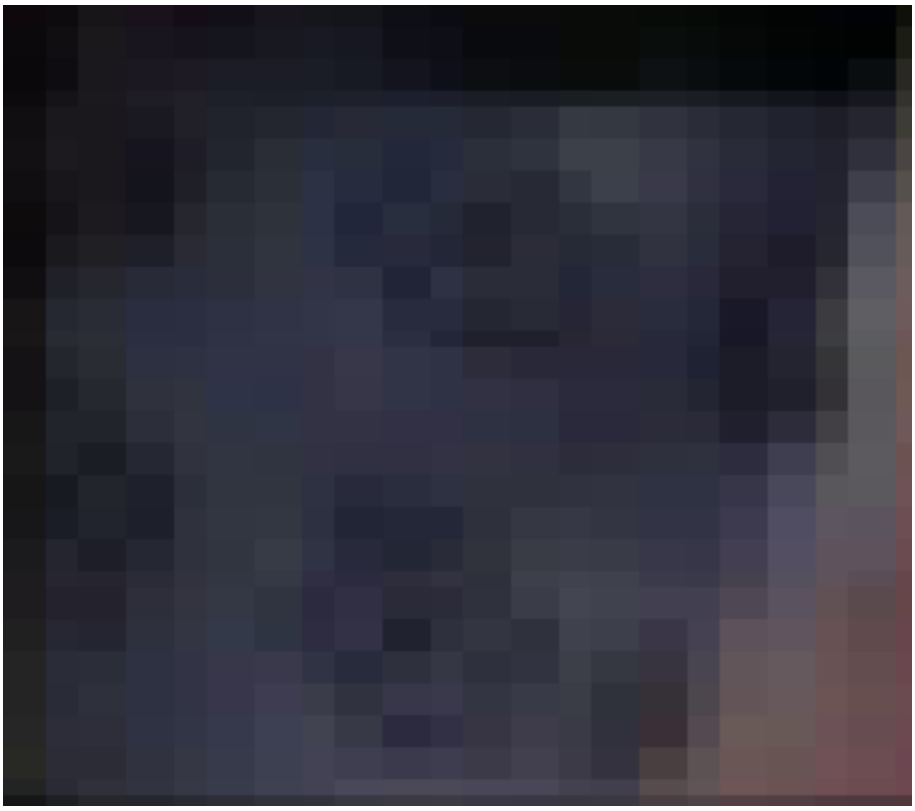


Fig.5.16 Imagen interpolación Michael_Netravali acercada al 600%

Con esta enterpolacion se intenta seraltar mas los detalles un como es el caso del punto del petalo de la flor (Fig.5.16) aunque no en todos los caso se consigue ya que como podemos obserbar la flores casi no se pueden observar

### Tiempo de Computo

El tiempo de computo de este algortimo aciende a 9 min a 10 min para generar 250 imágenes con lo que el promedia seria de 0,4 imágenes por segundo.

## 5.5. Bicubico Mejorado

Para este método hemos utilizado varios constantes de interpolación para poder y de esta forma viendo los resultados poder elegir la constante que mas nos interesa. Según las especificaciones técnica los valores recomendados para la constante de interpolación son entre

### Constante de Interpolación a=0

#### *Calidad de la Imgen*

Imagen con 100% de zoom (Fig.5.18):

Fig.5.18 Imagen interpolación Bic Mej a=0 acercada al 100%

Imagen con 300% de zoom(Fig.5.19)

Fig.5.19 Imagen interpolación Bic Mej a=0 acercada al 300%

Imagen con 600% de zoom(Fig.5.19)

Otra toma de la imagen con 600% de Zoom:

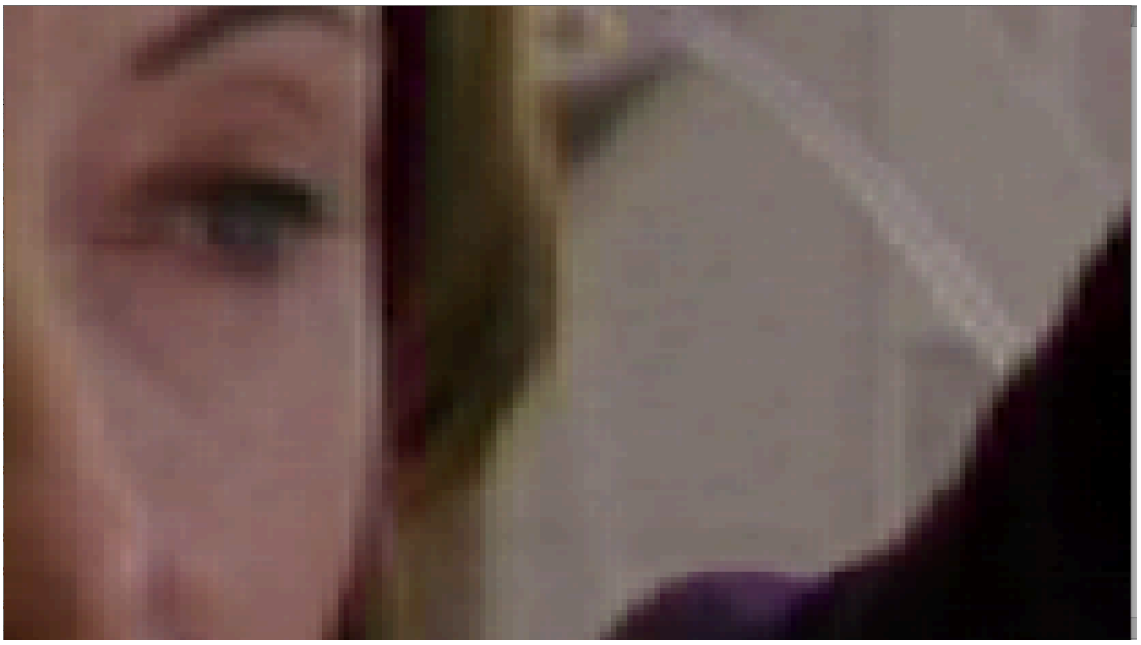
En esta imagen ya se puede observar una especie cruz griega con un punto en el centro aunque no hemos recuperado toda la información perdida con este algoritmo con lo que con este algoritmo es donde mas nos hemos acercado a la imagen original.

## Contante de interpolación a=0,5

Imagen al 100%

Imagen al 300%

Imagen al 600%



Otra Imagen con 600% es:

Visto los resultados nefastos que se obtienen con la constante de interpolación a=0,5 decidimos no incluirlo en el programa.

## Constante de interpolación a=-1

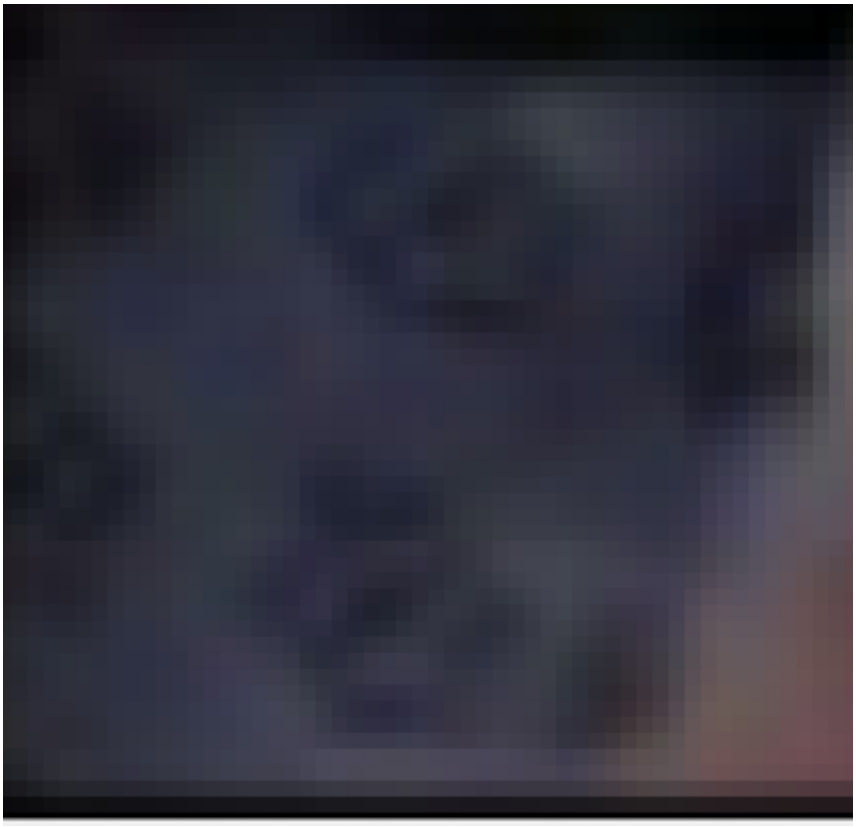
Por ultimo mostramos la imagen obtenida utilizando la interpolación bicubica mejorada pero esta vez con constante de interpolación a=-1

Imagen al 100%:

Imagen al 300%:

Imagen al 600%

Otra captura de la imagen al 600%:



Como se puede observar tanto en la forma de pétalo, en los tubos y en la retina de la mujer se obtienen aun mejores resultados que con la Vemos que con

esta constante se obtienen aun mejores resultados que con la constante a=0 con lo que la constante que se utilizara para este algoritmo es a=-1

### Tiempo de Computo

Dado que los tres algoritmos son exactamente iguales y solamente cambia la constante de interpolación por lo que el tiempo de computo es igual para las tres.

Para generar los 250 imágenes del tarda en realizarse de una media de 10 máximo 12 minutos con lo que el tiempo de computo seria una media de 0,35 Imágenes por segundo.

Este es el algoritmo que mas tiempo se necesita para la transformación de las imágenes pero en cambio es el que obtiene los mejores resultados

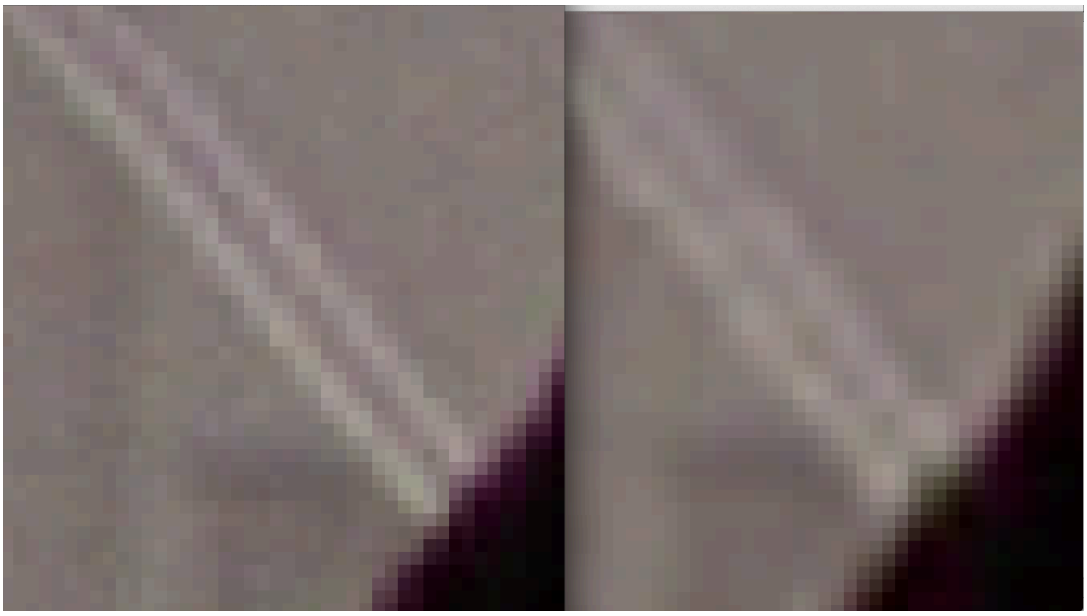
## 5.6. Diferencias entre algoritmos

Una vez expuestos los algoritmos vamos a compara alguno de ellos de tal forma que se pueda apreciar de forma mas clara la diferencia que puede haber entre los dos algoritmos

### Lineal y Bic. mejorado

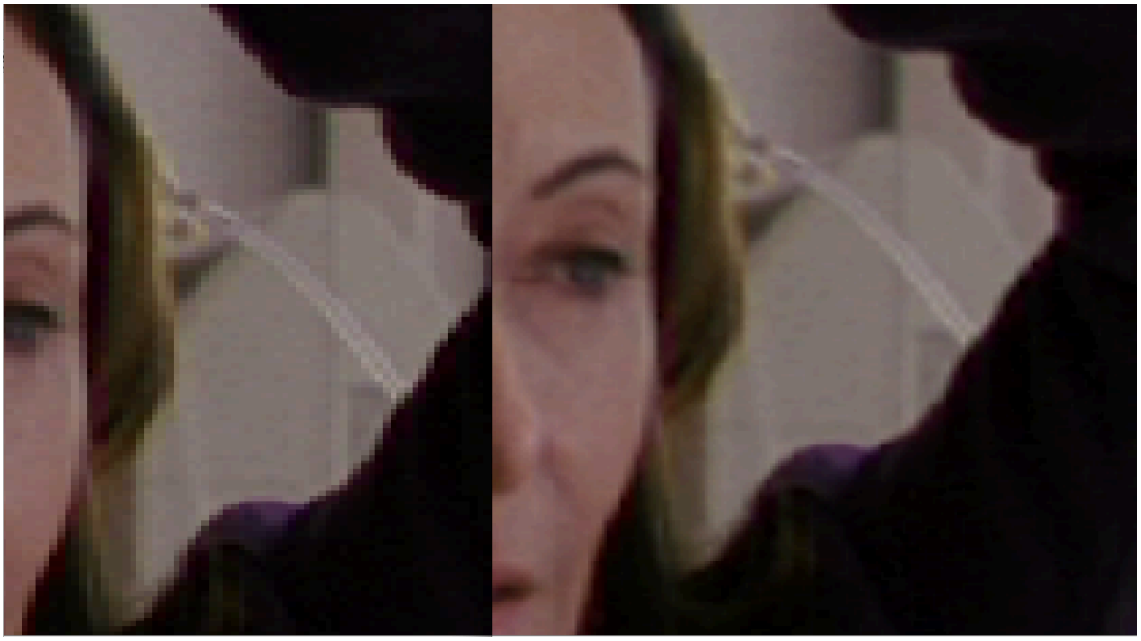
En esta imagen podemos ver como la imagen generada por el Bicubico Mejorado (la imagen de la derecha) muestra de una forma mucho mas clara de la presencia de los dos tubos(imagen de la izquierda).

## Imagen original y Bic. Mejorado



Como podemos observar la imagen original de HD se resaltan bastante mas los tubos que con la interpolación que se ha hecho con Bic. Mej. Pero esto ya es de suponer ya que no es posible mejorar la imagen original la cuestión es reducir al máximo la perdida de calidad.

## Lineal y Mitchell-Netravali



En esta captura vemos las diferencias que hay ente la imagen de Mitchell-Netravali (imagen a la izquierda) y la interpolación lineal en dosnde se puede observar que en la imagen de la izquierda se esta intentando resaltar los detalles pero a cambio se pierde mucho en el tema de nitidez.

# 6. Diseño y Guía de Usuario de la Herramienta

Para el diseño de la herramienta nuestro mayor objetivo ha sido crear una herramienta fácil de manejar y que requiera poco tiempo para poder entender el funcionamiento de esta:

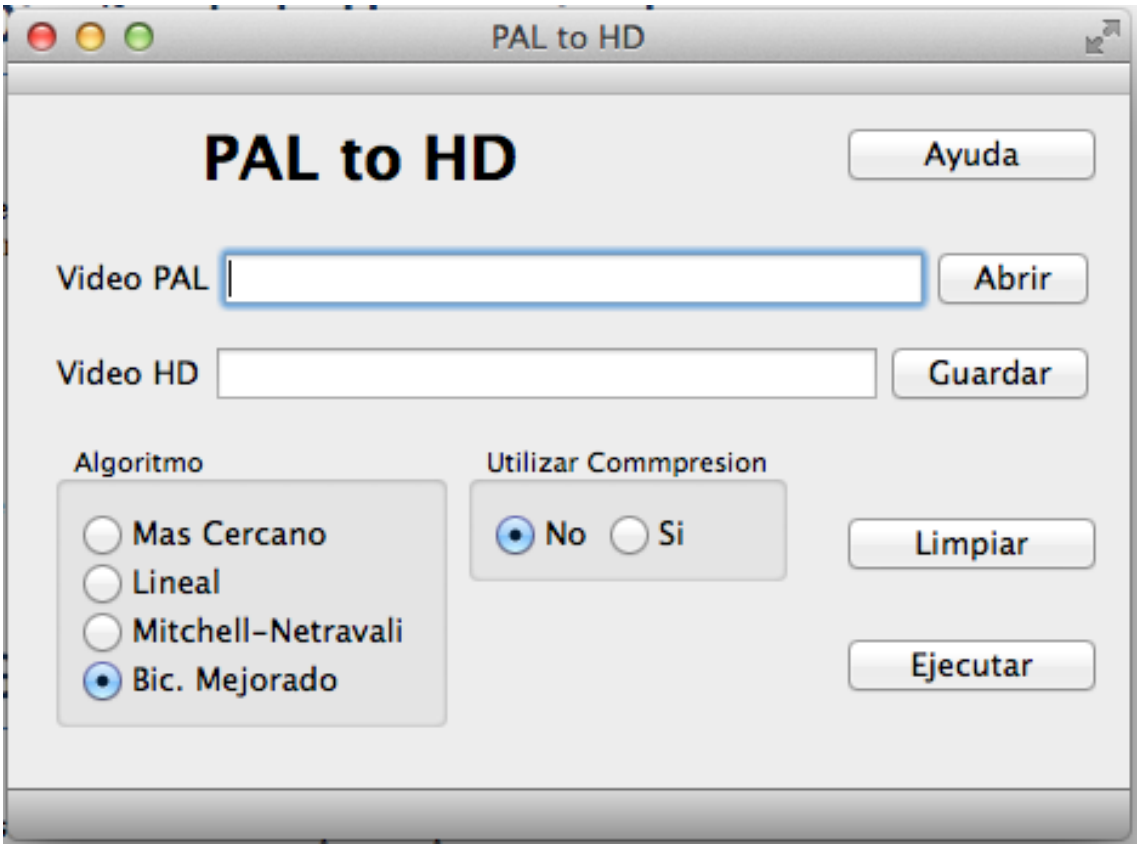



Fig.6.1 Programa Grafico

Esto es la parte grafica de la herramienta al arranque de esta a continuación se explicara el con mas detalle la función de cada una de las partes.

## 6.1. Botón Ayuda

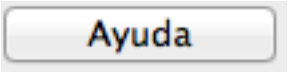



Fig.6.2 Botón ayuda

Este botón (Fig.6.2) esta ubicado en la parte superior derecha y tiene como función mostrarte una pequeña guía de usuario para ayudarte con el funcinamiento de la herramienta

## 6.2. Elegir Video PAL

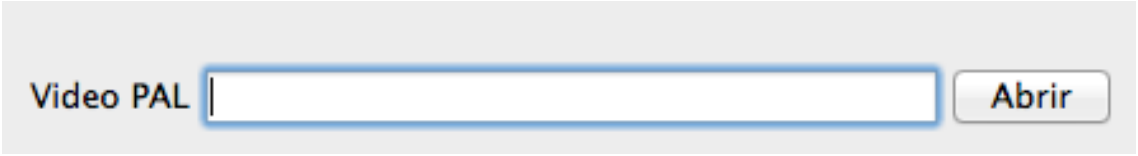



Fig.6.3 Campo Selección video Pal

Este campo (Fig.6.3) esta compuesto por tres parte por un lado tenemos el texto "Video Pal" y por otro tenemos una línea de Edición en la cual podemos seleccionar de forma manual el video o buscarlo mediante una forma mas interactiva (Fig.6.4) mediante la utilización del botón abrir.

Como por ejemplo:

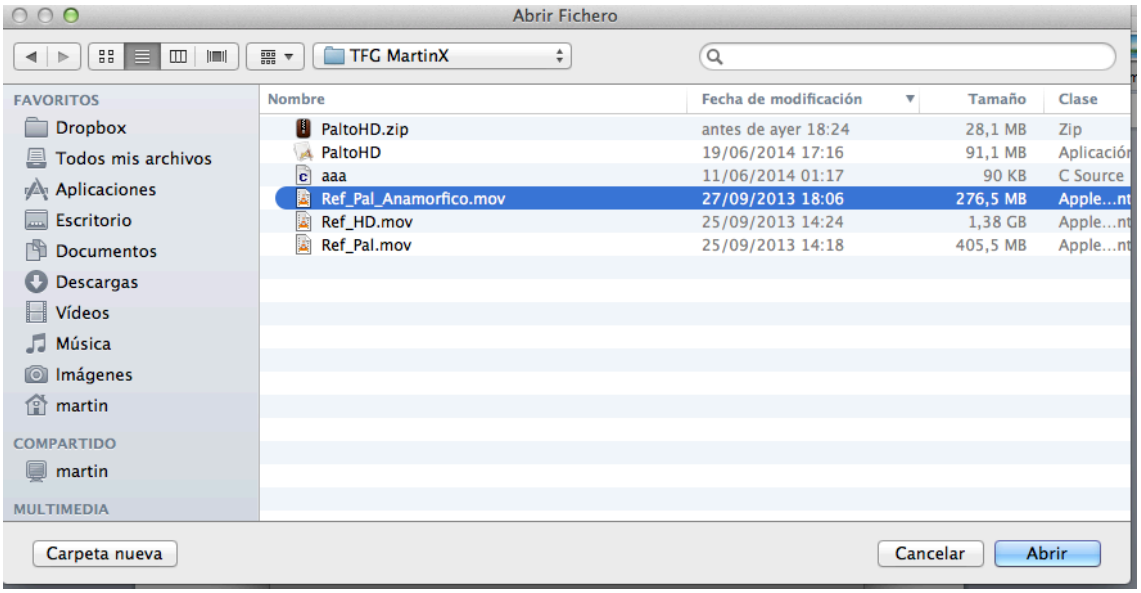



Fig.6.4 Buscar fichero Interactivo

Una vez seleccionado el fichero, nos sale un mensaje informándonos (Fig.6.5) de la ruta de selección del fichero:

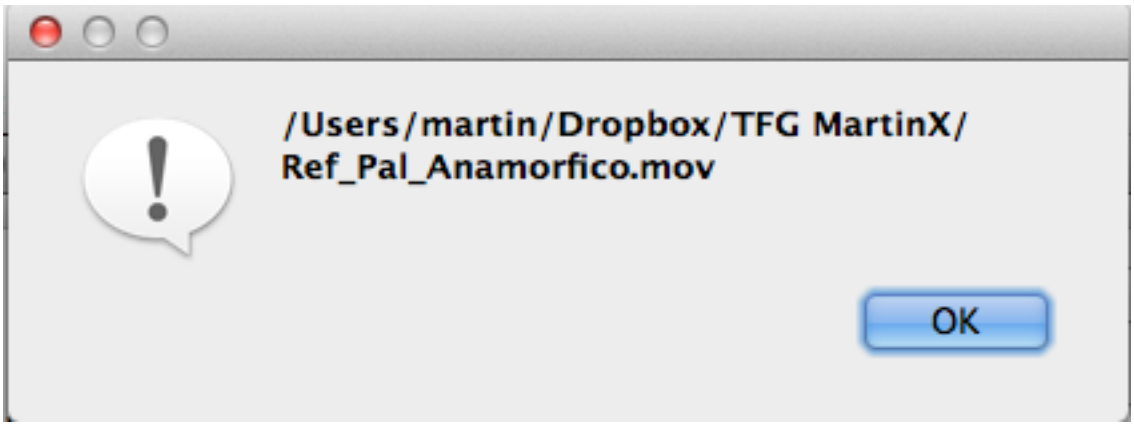



Fig.6.5 Mensaje Ruta Fichero

Una vez seleccionado el fichero, este nos lo mostrara en la línea de edición (Fig.6.6)

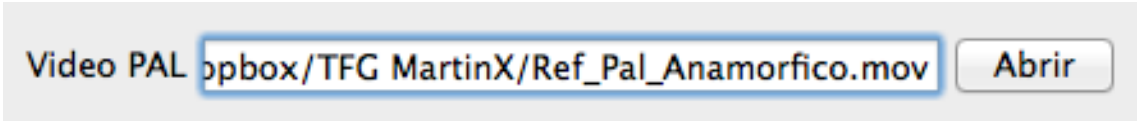



Fig.6.6 Línea edición Ruta PAL

Pudiendo de esta forma cambiar el nombre son la necesidad de abrir otra vez el botón abrir

## 6.3. Guardar Video HD

Para guardar el video en HD tenemos mas o menos la misma estructura que para seleccionar el Video PAL (Fig.6.7)


Fig.6.7 Campo Selección video HD

Tenemos 3 campos por un lado tenemos el texto “Video HD” y por otro tenemos la línea de edición en la cual podemos seleccionar la ruta de forma manual o por el contrario utilizar la forma interactiva que es pulsando el botón Guardar.

Al pulsar el botón guardar se nos abre la siguiente ventana:

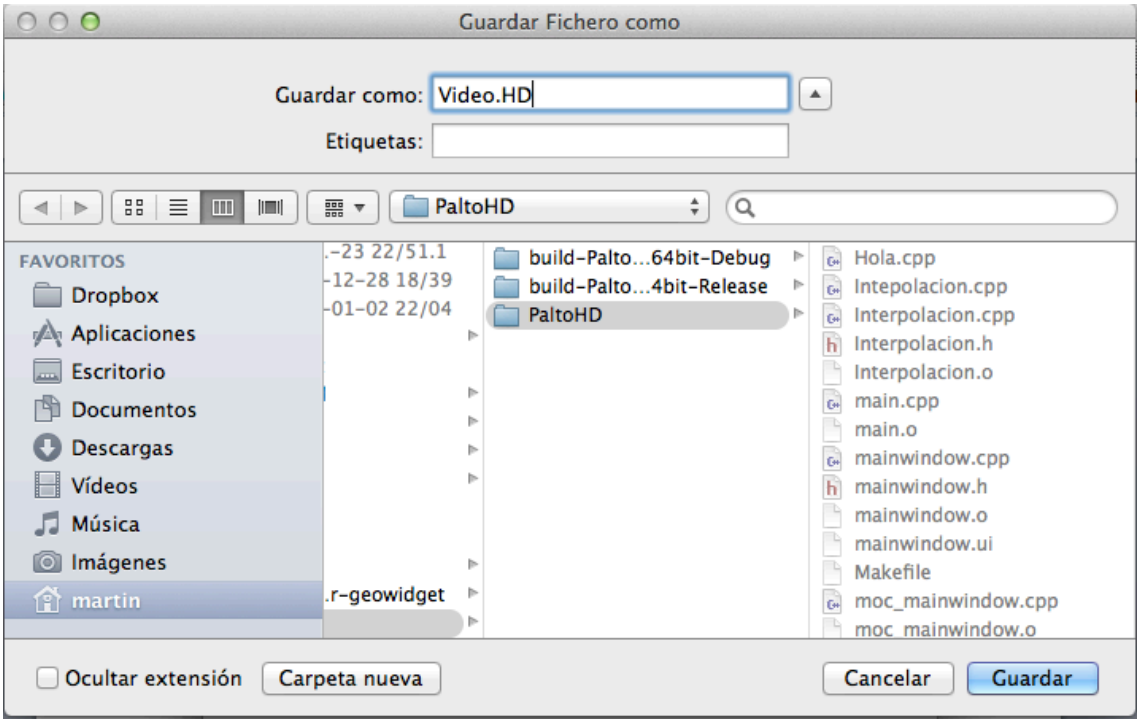



Fig.6.8 Guardar fichero modo interactivo

Donde como se puede observar podemos seleccionar de forma interactiva (Fig.6.8) tanto la carpeta donde queremos que se guarde el video como el nombre del video.

Una vez seleccionado el nombre si este nombre existe nos saldrá una ventana preguntándonos si queremos reemplazar el fichero (Fig.6.9).


Fig.6.9 Mensaje reemplazar fichero

Si seleccionas no nos deja en la ventana anterior pudiendo elegir otra ruta o nombre de fichero.

Una Vez seleccionada la ruta esta aparecerá en la línea de Edición(Fig6.10)

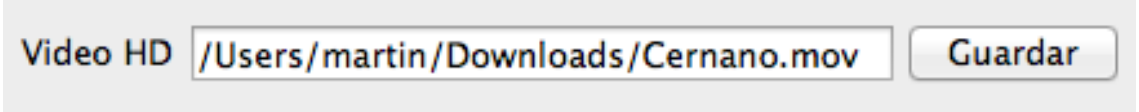


Fig.6.10 Línea edición Ruta PAL

Pudiendo modificar el nombre sin la necesidad de pulsar el botón de guardar para cambiar de nombre.

## 6.4. Algoritmo

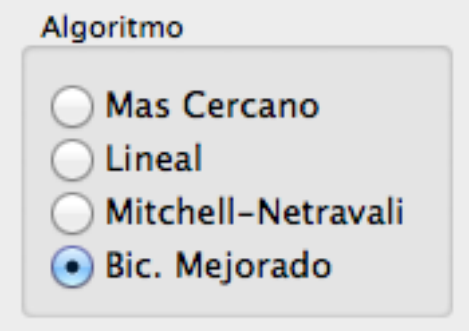



Fig.6.11 Selección Algoritmo

Para seleccionar el Algoritmo que deseamos implantar hemos creado esta caja (box) (Fig.6.11) en la cual vienen definidos todos los algoritmos que hemos implementado. Estos algoritmos son excluyentes con lo que es imposible seleccionar dos algoritmos a la vez. Al iniciar el programa hemos definido que el algoritmo por defecto sea "Bicubico Mejorado"

## 6.5. Compresión

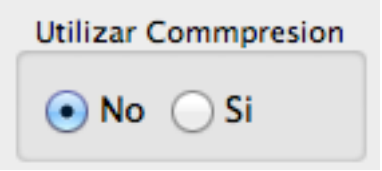



Fig.6.12 Selección Compresión

Al igual que en Algoritmo para la Compresión hemos creado otra caja (box) (Fig6.12) en la cual podemos elegir entre la utilización de compresión, quedando las dos opciones excluyentes una de otra.

## 6.6. Limpiar

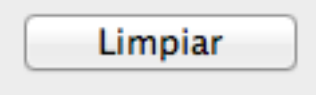



Fig.6.13 Botón Limpiar

Este botón sirve para limpiar (Fig.6.13) los campos de la línea de Edición tanto del video pal como del video HD.

Partiendo de (Fig.6.14)

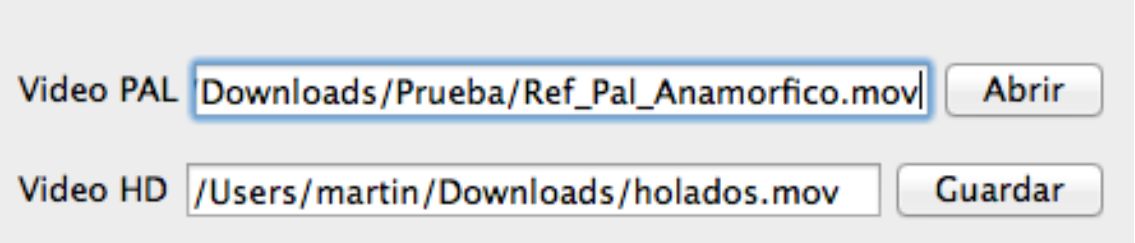



Fig.6.14 Campos Líneas de Edición Escritas

al pulsar el botón (Fig.6.13) estos dos campos se limpian quedando de la siguiente manera (Fig.6.15)

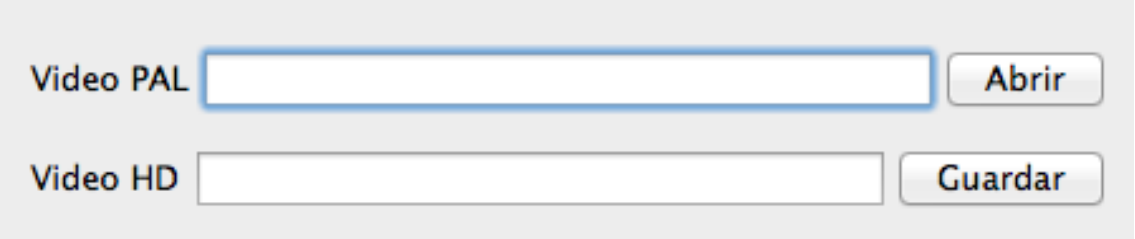



Fig.6.15 Campo líneas de Edición Vacías

## 6.7. Ejecutar

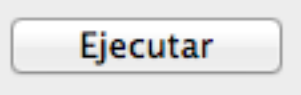



Fig.6.16 botón Ejecutar

Este es el botón (Fig.6.16) encargado de lanzar la interpolación una vez que tenemos puestos el video y la ruta donde queremos guardarlo en el caso en el cual no ponemos algunos de estos campos nos saldrán los siguientes mensajes(Fig.6.17) y (Fig6.18)

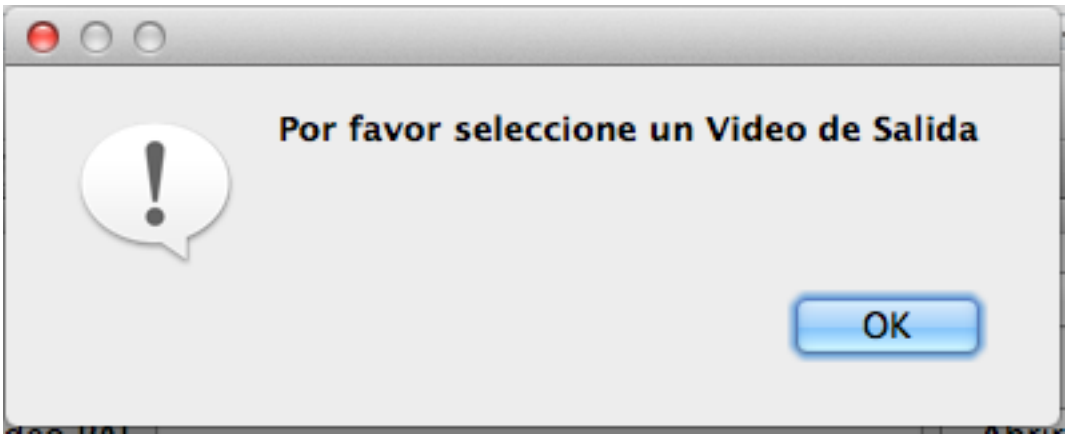



Fig6.17 Error Video PAL no Seleccionado


Fig6.18 Error Se se ha seleccionado Ruta para guardar el Video HD

En los caso que no hayamos seleccionado el video de entrada o no hayamos puesto una ruta para guardar el video.

Una vez finalizada la tarea salta un mensaje avisando de esto(Fig.6.19)

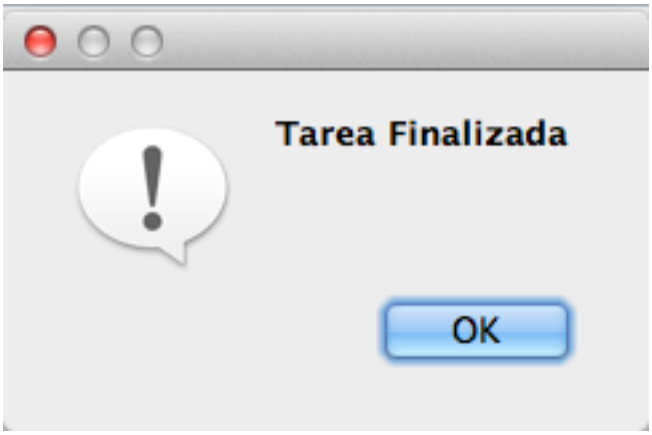



Fig.6.19 Tarea finalizada

# 7. Bibliografía


- [1] http://www.ffmpeg.org/
- http://www.deetc.isel.ipl.pt/Analisedesinai/sm/downloads/doc/ch08.pdf
- http://tech.ebu.ch/docs/techreview/trev_300-wood.pdf
- http://www.digicamsoft.com/bmp/bmp.html
- http://www.codeproject.com/Articles/236394/Bi-Cubic-and-Bi-Linear-Interpolation-with-GLSL
- https://trac.ffmpeg.org/wiki
- http://members.bellatlantic.net/~vze2vrva/design.html
- http://qt-project.org/
- http://pixinsight.com/forum/index.php?topic=559.0

Este documento esta firmado por

| | | |
|---|---|---|
|  | **Firmante** | CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES |
| | **Fecha/Hora** | Fri Jun 27 00:16:49 CEST 2014 |
| | **Emisor del Certificado** | EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES |
| | **Numero de Serie** | 630 |
| | **Metodo** | urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature) |